Inlupp 1: Utskrifter och loopar

Imperativ Programmering DT501G, HT 2019
19 november 2019

I den första labuppgiften kommer du att bekanta dig med och få övning i den nödvändigaste syntaxen i C och hur den skiljer sig från Python, och få övning i grundläggande hantering av variabler och flödeskontroll, samt input och output från program.

Läs igenom de generella instruktionerna först (i separat dokument på Blackboard), innan du tar dig an den här uppgiften!

Fråga 1 Har du läst och förstått de generella instruktionerna?

Deadline

Deadline för inlämning är **27 november**.

Lärandemål

Målet med uppgifterna i den här laben är att du ska lära dig behärska följande moment:

- grundläggande text-inmatning och utmatning (till och från `stdin` och `stdout`),
- läsa och skriva data i variabler (heltal och reella tal),
- kommentarer i programkod,
- grundläggande flödeskontroll (**`if`**/**`else`** etc),
- algoritmiskt tänkande: att kunna skriva en algoritm för att lösa ett mindre problem.

Uppgifter

1.1 Skriv ditt namn

Skriv ett C-program som skriver ut ditt namn, din e-postadress och var du kommer ifrån på `stdout` enligt följande format (med varje post på en egen rad).

Till exempel:

```
Martin Magnusson
martin.magnusson@oru.se
Karlskoga
```

Se till att ditt program returnerar 0 när det avslutas.

1.2 Räkna med cirklar (1)

Skriv ett C-program som beräknar arean och omkretsen av en cirkel med radien $r = 1$, och skriver ut resultatet.

Du ska använda ekvationerna nedan för din beräkning, även om resultatet alltid blir samma. (Det är alltså inte en lösning att räkna ut svaret i förväg och skriva ut en fördefinierad textsträng!) Du kan approximera π med 3.1415 för den här uppgiften.

$$A = \pi r^2 \qquad O = 2\pi r$$

Skriv ut resultatet på två rader, enligt nedan. (Antal decimaler i utskriften spelar ingen roll i det här fallet, så länge det är minst två.)

```
Area 3.14
Omkrets 6.28
```

1.3 Räkna med cirklar (2)

Skriv ett C-program som beräknar arean och omkretsen av en cirkel som ovan, men som läser in radien från användaren. (Nu kommer du alltså att behöva använda även funktionen `scanf`, utöver `printf`.)

```
$ ./task_3
Radie? 10
Area 314.15
Omkrets 62.83
```

1.4 Loop

Skriv ett program som, med hjälp av en **`for`**-loop, skriver ut alla heltal från och med 1 till och med 20, separerade med mellanslag.

1.5 Loop/2

Skriv ett program som, med hjälp av en **`for`**-loop, skriver ut alla tal från och med 1 till och med 20 som är jämnt delbara med antingen 2 eller 5. Separera talen med mellanslag. (Här behöver du använda modulo-operatorn `%` samt OR-operatorn `||`.)

1.6 Loop de loop

Skriv ett program som tar två heltalsargument, ett för timmar och ett för minuter, och skriver ut klockslaget var tionde minut från kl 00:00 och fram till det angivna klockslaget. Läs in var sitt heltal för timmar och minuter med `scanf`.

```
$ ./task_6
Skriv in timme och minut: 1 15
00:00
00:10
00:20
00:30
00:40
00:50
01:00
01:10
```

För att skriva ut tal tvåsiffrigt (alltså ”02” i stället för ”2”) använder du specifikationen `%02d` till `printf`. (Se även kapitel 2.4.)

Du ska använda *nästlade* loopar i din lösning.

Du behöver inte hantera fall där någon skriver in en timme som är större än 23 eller ett minutvärde som är större än 59, eller skriver in något annat än siffror.

1.7 do

Skriv ett program som gör samma sak som i uppgift 1.6, men med **`do`**-loopar i stället.

1.8 while

Skriv ett program som gör samma sak som i uppgift 1.6, men med **`while`**-loopar i stället.

1.9 Slumptal

Skriv ett program som simulerar ett tärningskast (med en sex-sidig tärning). Varje gång programmet körs ska det helt enkelt skriva ut ett slumpvis valt tal mellan 1 och 6.

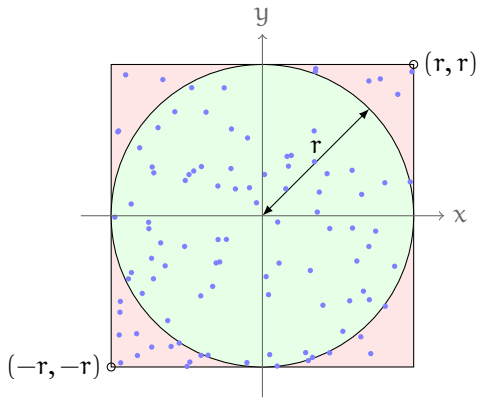
För att generera slumptal kan du använda funktionen `rand` som du får tillgång till med **`#include`** `<stdlib.h>`.[1] Funktionen `rand` returnerar ett slumpvis valt heltal mellan 0 och konstanten `RAND_MAX`, som är lite olika i olika C-implementationer. För att få ett värde mellan 1 och 6 kan du använda modulo-operatorn `%`. (Ett heltal – vilket som helst – modulo 6, blir ju alltid mellan 0 och 5!) För att få *olika* slumpvärden behöver du också anropa `srand`, som sätter ett ”slumpfrö”. För att få olika slumpvärden *varje gång* behöver du också välja frö från en källa som ändrar sig; t.ex. klockan. Du kan anropa `time(NULL)` från time.h får att få ett nytt frö varje sekund.

1.10 Räkna med cirklar (3)

Skriv ett C-program som beräknar arean av en cirkel, men nu med hjälp av så kallad *Monte Carlo-integrering*. Precis som ovan ska du läsa in radien från användaren.

Monte Carlo-integrering är ett sätt att numeriskt beräkna en integral (=area), som är användbar för väldigt knöliga funktioner. Arean av en cirkel är visserligen inte knölig, men med en så pass lätt funktion är det lätt att kontrollera att man har gjort rätt.

För att använda Monte Carlo-integrering så behöver du först bestämma ett intervall som du ska integrera över. För att få med hela cirkelns area behöver intervallet vara minst så stort som cirkeln. Ett vettigt intervall i det här fallet är då den kvadrat som innesluter cirkeln. Om du har en cirkel med mittpunkt $(0, 0)$ och radie $r = 1$ så får du då en kvadrat som går mellan hörnen $(-1, -1)$ och $(1, 1)$. Figuren nedan illustrerar det hela. Funktionen som vi vill integrera har värdet 1 i det gröna intervallet och 0 överallt annars.



Algoritmen för Monte Carlo-integrering ser ut så här:

1. Initiera en variabel för totalsumma: $S \leftarrow 0$.

[1] Du kan läsa mer om `rand` och andra funktioner från standardbiblioteken på The C Library Reference Guide.

2. Initiera en variabel för skattning av arean: A.
3. Initiera en räknare: $i \leftarrow 0$.
4. Iterera följande tills du tror dig ha uppnått tillräcklig noggrannhet.
   (a) Stega upp räknaren: $i \leftarrow i + 1$.
   (b) Generera ett slumptal inom intervallet. (För en kvadrat behöver du generera en tvådimensionell koordinat, det vill säga ett slumpvärde för x och ett för y.)
   (c) Beräkna värdet av funktionen som ska integreras för den genererade punkten. I vårt fall har funktionen värdet 1 om punkten ligger innanför cirkeln, och 0 annars. Här måste du alltså kolla om din slumpvis valda punkt (x, y) ligger innanför cirkeln; med andra ord, om $\sqrt{x^2 + y^2} \leqslant r$.
   (d) Multiplicera värdet du fick med arean för integreringsintervallet (det vill säga kvadraten); alltså med $(2r)^2$.
   (e) Addera värdet till totalsumman som du räknar upp.
   (f) Ändra den nuvarande uppskattningen av arean till $A \leftarrow S/i$.
5. Returnera A.

I den här uppgiften du alltså dra slumpvis valda flyttal i stället för heltal. För att få ett värde mellan 0 och 1 kan du dividera talet du får från `rand` med konstanten `RAND_MAX`. (Kom ihåg att se till att det blir flyttalsdivision, och inte heltalsdivision.)

Du kommer också behöva använda (minst) en annan funktion från biblioteket `math.h`. Funktionen `sqrt` därifrån beräknar kvadratroten av ett tal (som ju behövs för att veta om en punkt är inuti en cirkel eller ej). I `math.h` finns även funktionen `fabs` som är praktisk för att få absolutvärdet av ett flyttal och `pow` för att beräkna potenser. Förutom att inkludera math.h behöver du också *länka* med math-biblioteket. Vi kommer att prata mer om det lite senare i kursen, men just nu räcker det med att veta att du måste lägga till `-lm` på kommandoraden när du ska kompilera och bygga ditt program. Alltså: `gcc task_11.c -Wall -std=c99 -lm -otask_11`

Vad innebär det då att iterera ”tillräckligt” länge? Dels kan du ju sätta ett maxtak på `i`, men du ska också sätta ett stoppkriterium som beror på hur mycket A har ändrats. När den uppskattade arean bara ändrar sig med någon decimal långt bort så kommer den förmodligen inte att ändras så mycket mer även om du itererar längre i loopen.

Tanken med den här uppgiften är att öva mer på loopar, med mer avancerat stoppkriterium än en fast gräns, samt att prova på slumptalsgenerering och funktioner från `math.h`.

Fundera över de här frågorna. (Du behöver inte redovisa dina svar.) Hur länge behöver du iterera innan du får ungefär rätt värde på arean när $r = 1$ respektive $r = 2$? Hur många decimalers noggrannhet får du?

1.11 The final countdown

Skriv ett program som skriver ut en förloppsindikator (”progress bar”) som räknar ned hur mycket som är kvar av en viss process, från 100 % till 0 %. Din algoritm ska använda en parameter som bestämmer hur många steg som skrivs ut. Läs in värdet på parametern med `scanf`. Se exempel nedan.

Skriv ut dels hur många procent som är kvar, och skriv också ut en rad med stjärnor. För varje helt tiotal procent som är kvar ska en stjärna skrivas ut. I första steget skrivs alltid 10 stjärnor ut.

Tips: du kommer antagligen att använda en loop inuti en annan loop.

```
$ ./task_11
Hur många steg? 10
[ 100 av 100 kvar] * * * * * * * * * *
[  90 av 100 kvar] * * * * * * * * *
[  80 av 100 kvar] * * * * * * * *
[  70 av 100 kvar] * * * * * * *
[  60 av 100 kvar] * * * * * *
[  50 av 100 kvar] * * * * *
[  40 av 100 kvar] * * * *
[  30 av 100 kvar] * * *
[  20 av 100 kvar] * *
[  10 av 100 kvar] *
[   0 av 100 kvar]
```

```
$ ./task_11
Hur många steg? 3
[ 100 av 100 kvar] * * * * * * * * * *
[  67 av 100 kvar] * * * * * *
[  34 av 100 kvar] * * *
[   1 av 100 kvar]
```

Frivilligt **1.12 Uppgifter från boken**

Om du har gjort klart uppgifterna ovan och vill få mer övning så kan du gärna också göra uppgifterna i kapitel 1–4 i ”C från början”.

Frivilligt **1.13 LOL**

Om du vill ha mer utmaningar kan du också, för skojs skull, testa att implementera uppgifterna i programmeringsspråket LOLCODE. Det är också ett renodlat imperativt språk, om än gjort bara för att det är kul, och inte för att skapa några större program i... Däremot är det en bra övning att uttrycka samma program (semantik) med olika syntax. Du kan ladda ned en programtolk för LOLCODE, och hitta dokumentation, på `http://lolcode.org/`. Fråga Martin om du vill veta mer om detta.

```
HAI 1.2
CAN HAS STDIO?
VISIBLE "HAI WORLD!!!1!"
KTHXBYE
```

elistmania.com